Proyecto de Sistemas Informáticos 2008-2009

Facultad de Informática

Universidad Complutense de Madrid

Geomantes:

Sistema de navegación GPS por mapas gráficos

Antonio Ariza Guerrero

Javier Doria Dulanto

Isamu Takebe Heras

Profesor director: Fernando Sáenz Pérez

INDICE

RESUMEN

1. RESUMEN EN CASTELLANO

En esta memoria describimos todo el trabajo realizado como proyecto para la asignatura Sistemas Informáticos de la Ingeniería Superior en Informática de la Universidad Complutense de Madrid. Geomantes pretende ser una solución gratuita para aquellos usuarios de telefonía móvil que busquen un navegador GPS por mapas gráficos (*raster* o *bitmap*, ej. OziExplorer) intuitivo pero a la vez completo.

La elaboración de *Geomantes* nunca ha perdido de vista las dos premisas sobre las que fue concebido: **gratuidad** y máxima **compatibilidad** posible. Se ha trabajado siempre con tecnologías gratuitas y hemos evitado adquirir licencias. Para lo segundo, se han extendido las funcionalidades a todos los tipos de terminales móviles que han sido técnicamente posibles.

A lo largo del desarrollo de la aplicación nos hemos encontrado con una serie de retos que hemos superado satisfactoriamente. Estas dificultades son las que han ido moldeando la forma, y en cierta medida, los contenidos que en esta memoria se explicarán.

- Daremos una explicación detallada de las características de la aplicación, ilustrándolo con diagramas.
- Enumeraremos los casos de uso.
- Hablaremos de las especificaciones y requisitos mínimos.
- Detallaremos a fondo todas las tecnologías involucradas en el desarrollo.

Por último, elaboraremos una conclusión de si los objetivos logrados se ajustan a los que nos planteamos inicialmente, resaltaremos los puntos fuertes (y los débiles) y trataremos una serie de posibles mejoras que harían de *Geomantes* una aplicación más interesante.

2. ENGLISH SUMMARY

In this report we describe all the work carried out in the final project for the Computer Systems course of Computer Engineer Studies at the Complutense University of Madrid. Geomantes wants to serve as a free alternative for all those users of mobile phones who are looking for an intuitive graphic map GPS navigator that is also complete.

The development of Geomantes never lost sight of the two premises on which it was created: **free use** and **maximum compatibility**. To achieve the first we only worked with free technology and avoided the acquisition of licenses. For the second premise, we extended the functions to all types of mobile devices, which turned out to be technologically feasible.

Throughout the development of the application we came across a series of challenges that we were able to surpass adequately. These difficulties have shaped the methods and, to some extent, the contents which are explained in this report.

In the report we are going to:

- Give a detailed explanation of the characteristics of the application, illustrating it with UML diagrams.
- List the use cases.
- Talk about the specific details and the minimum requisites for the application.
- Thoroughly detail all the technologies involved in its development. Among these technologies, those required for the connection between the mobile device and the GPS as well as the Bluetooth functions are especially noteworthy.

Finally, we are going to form a conclusion regarding whether or not the goals that we reached correspond with the original ones. We will also highlight the strong points (and weak points) and we will discuss a series of upgrades that would make Geomantes a more interesting application.

DESCRIPCIÓN DEL PROYECTO

1. MOTIVACIÓN

El mercado de aplicaciones GPS para móviles es extenso tanto en oferta de aplicaciones como cantidad en posibilidades. A pesar de esto, Geomantes nace ante la motivación de unificar las cualidades más interesantes de productos que se encuentran ya en el mercado y con el aliciente de ser gratuito.

Se observan dos enfoques muy diferenciados en las aplicaciones de navegación. El ser un ámbito muy especializado hace que un gran número de programas estén orientados al mercado profesional. Desgraciadamente, por norma general, presentan interfaces poco elaboradas y las posibilidades, aunque muy concretas, resultan limitadas para un usuario de a pie. Por otra parte, podemos agrupar muchas de las aplicaciones GPS restantes como orientadas " al gran público ", descuidándose de manera injustificada las posibilidades y aplicaciones de la aplicación para que la experiencia de usuario resulte simple y eficiente.

Por lo tanto Geomantes trata de cumplir con las expectativas de todo tipo de usuarios, elevando los ámbitos de uso de la aplicación a lo que permiten las tecnologías disponibles hoy en día.

2. FILOSOFÍA SUBYACENTE

Desde el momento de su concepción, el proyecto de Geomantes mantiene inalterable las premisas, como decíamos, de la gratuidad y la máxima compatibilidad.

Somos conscientes de que no podemos hacer frente a grandes empresas con modelos como TomTom, Garmin, Panasonic o Navman. Es por ello que, desde que comenzamos a concebir este programa, nos dimos cuenta que teníamos que potenciar otras cualidades con las que distanciarnos (o incluso aventajarnos) de las anteriormente mencionadas. El que Geomantes sea OpenSource y gratuito, previsiblemente atraerá a un número de usuarios que no se pueden permitir o no desean costearse el precio de un GPS comercial.

Hemos sido cuidadosos a la hora de elegir la plataforma de desarrollo. La elección de Java nos permite alcanzar un número altísimo de potenciales usuarios al ser el referente en desarrollo de aplicaciones móviles con J2ME. Otras como Windows Mobile o Symbian son igualmente atractivas, pero excluyentes.

3. OBJETIVOS

Hemos planteado como objetivos, además de los ya enumerados, los siguientes:

- Geomantes ha de ser un programa útil para todos los usuarios, desde los que buscan en él un simple sistema de localización hasta los que lo usarán como navegador offroad.
- Se proporcionará una extensa y detallada documentación para que futuros desarrolladores puedan expandir el trabajo realizado y completar un abanico de funcionalidades más amplio.
- Brindar a usuarios sin un móvil de última generación, o por ejemplo sin GPS, la posibilidad de usar una parte de las características del programa. La navegación “ offline ” para el recorrido de rutas ya trazadas es un ejemplo de ello.

4. HISTORIAL DE VERSIONES

15 de noviembre de 2008 – Versión 0.1

Primera versión de Geomantes en la que se desarrollo lo estrictamente necesario para comunicar la aplicación con un dispositivo GPS a través del móvil.

15 de diciembre de 2008 – Versión 0.2

Extensión de las funcionalidades GPS para permitir dispositivos Bluetooth que se encuentren al alcance. Además, se desarrolla la posibilidad de funcionamiento sin GPS a través de una ruta ya realizada. Para ello es necesario implementar una forma de navegación por el sistema de archivos del móvil.

30 de enero de 2009 – Versión 0.3

Primera aproximación a la geolocalización de coordenadas en los mapas. Pese a no ser perfecta, resulta muy eficaz para continuar con el desarrollo.

30 de febrero de 2009 – Versión 0.4

Depuración del sistema de geolocalización y desarrollo de una modalidad de navegación sobre Google Maps.

25 de marzo de 2009 – Versión 0.5

Implementación de un sistema de opciones permanente en memoria. Depuración del interfaz de usuario. Se comienza a trabajar con la posibilidad de contar con puntos de interés.

15 de mayo de 2009 – Version 0.6

Desarrollo de un interfaz nuevo, más intuitivo y gráfico que el anterior. El sistema de POI alcanza un alto porcentaje de funcionalidad, permitiéndose su inserción por parte del usuario y el filtrado por distintos tipos.

5 de julio de 2009 – Versión 0.7

Implementación de la interfaz y su conexión con los distintos módulos de la aplicación. Ampliación de las prestaciones de la aplicación y subsanación de errores. Se decide prorrogar el periodo de desarrollo de la aplicación hasta septiembre para poder completar satisfactoriamente todo lo que tenemos planeado y tener un plazo de pruebas considerable.

20 de agosto de 2009 – Versión 0.9

Se terminan de implementar las funcionalidades previstas y se comienza a realizar un extenso plan de pruebas.

15 de septiembre de 2009 – Versión 1.0

Concluye el periodo de pruebas y el programa alcanza su primera versión estable y apta para su distribución.

5. ESPECIFICACIONES TÉCNICAS

Las herramientas utilizadas para el desarrollo han sido:

- *Java ME SDK 3.0 (http://java.sun.com)*
  Herramienta de desarrollo de software en Java para PC.

- *Sun Java Wireless Toolkit 2.5.2 (http://java.sun.com/products/sjwtoolkit/).*
  Entorno de desarrollo de software en Java para móviles.

- *Nokia S60 5th Edition SDK para Symbian OS*
  (http://www.forum.nokia.com/info/sw.nokia.com/id/ec866fab-4b76-49f6-b5a5-af0631419e9c/S60_All_in_One_SDKs.html)
  El SDK habilita el desarrollo en Symbian C++, Open C/C++, Java y diversas tecnologías web. El SDK está basado en S60 y Symbian OS 9.4. Incluye todos los recursos necesarios para el desarrollo de aplicaciones (documentación, API, ejemplos y emulador), excluyendo el IDE.

- *OziExplorer (http://www.oziexplorer.com/)*
  Es un sistema de navegación ‘ offroad ’ caracterizado porque no es vectorial al contrario que otros programas como TomTom. OziExplorer no permite indicar la posición del vehiculo dentro de las vías al no ser un programa de cartografía vectorial. TomTom asignará nuestra posición al camino más cercano mediante una especie de “ imantación ” . OziExplorer, en cambio, al hacer uso de cartografía en imagen o RASTER, navegará a través de imágenes cartográficas (u ortomapas) indicándonos la posición exacta que marca la señal GPS.

- *Adobe Photoshop CS4 (http://www.adobe.com/es/products/photoshop).*
  Proporciona las utilidades necesarias para realizar los retoques fotográficos pertinentes.

- *NetBeans IDE 6.1.*
  Entorno de desarrollo integrado (IDE) para crear aplicaciones de cualquier tipo. La experiencia positiva con este entorno conferida por un interfaz intuitivo, las facilidades que se plantean a la hora de programar y la multitud de funcionalidades gráficas presentes, han hecho que nos decantásemos por este.

- *CollabNet Subversion Server 1.5.3-1.*
  Utilidad para mantener comunicación entre los desarrolladores y mantener actualizada la aplicación.

- *Google Code (http://code.google.com/).*
  El repositorio para código fuente de Google nos ha servido como lugar de almacenaje de todas y cada una de las versiones de nuestra aplicación.

- *Microsoft Word.*
  Herramienta de procesamiento de textos.

- *ProGuard (http://proguard.sourceforge.net/).*
  Herramienta de ofuscación de código Java.

TECNOLOGÍAS EN USO

1. DESCRIPCIÓN

El número de retos que se han debido de afrontar ha sido numeroso. El principal contratiempo era la falta de experiencia en desarrollo para móviles con J2ME, y más tratándose de una aplicación que gira en torno a la conectividad GPS, nunca trabajada en nuestra experiencia.

Para poder compatibilizar la aplicación con el mayor número de dispositivos, nos encontramos frente al contratiempo de tener que escribir distintos códigos para según qué fabricante ejecutase la aplicación.

2. JSR 179: Location API *[3]*

JSR 179 son las siglas detrás del API de localización disponible para J2ME. A través de este paquete se nos brinda acceso a las funcionalidades más comunes de los dispositivos GPS. Permite una simple conexión entre nuestra aplicación y susodicho dispositivo, permitiendo al programador despreocuparse de determinados aspectos técnicos. Para entender cómo funciona el API, basta con enumerar sus componentes fundamentales:

`javax.microedition.location.Criteria:` Proporciona funciones que nos permiten filtrar los proveedores de localización disponibles. Podemos tener en cuenta parámetros como la precisión horizontal y vertical, el que el dispositivo sea de pago o el nivel de consumo de batería.

`javax.microedition.location.LocationProvider:` Es el punto de partida para usar el API. Representa el modulo proveedor de localizaciones, es decir, el GPS. Podemos obtener su estado y su localización en coordenadas.

Una vez definidos los criterios, obtendremos una referencia al proveedor de contenidos. La clase LocationProvider implementa un método estático getInstance(Criteria criteria) que devuelve un proveedor que se adapte a los criterios que hemos indicado. Si ninguno de los proveedores disponibles encaja en los criterios indicados, el método devolverá null. En caso de que no exista ningún proveedor de localización disponible se lanzará una LocationException. Es posible llamar a getInstance pasando null como parámetro: de esta forma se especifican los parámetros menos restrictivos para la selección.

Utilizaremos la instancia obtenida de LocationProvider para obtener la posición mediante el método getLocation. En el podemos especificar un *timeout*, o tiempo que queremos que espere el sistema para obtener una posición. Una vez obtenida dispondremos de un objeto Location que contendrá, entre otras cosas, las coordenadas de nuestra ubicación.

`javax.microedition.location.Coordinates`: Tipo de objeto usado por el Location API que sirve para describir las coordenadas que emite el dispositivo GPS. Estas son almacenadas como datum WGS84. La latitud y longitud son valores expresados en grados decimales (en vez de minutos/segundos).

De forma opcional, dependiendo del dispositivo, podremos obtener información de altitud (con su precisión), velocidad y dirección.

Por último, es importante comprobar que el API JSR 179 está implementado en el dispositivo en el que se ejecuta la aplicación. Para ello es necesario comprobar que la clave correspondiente a "`microedition.location.version`" al ser llamado "`System.getProperty`" devuelve "`1.0`".

3. JSR 82: Bluetooth API

JSR-82 es un estándar definido por el *Java Community Process* para proveer un estándar al desarrollo de aplicaciones Bluetooth en Java. El API JSR-82 oculta la complejidad del protocolo Bluetooth mediante la exposición de un puñado de sencillas funciones.

Los pre-requisitos para poder hacer uso de este API son un dispositivo compatible y un dispositivo Bluetooth.

Gracias a JSR-82 podremos realizar las siguientes acciones:

1. Manejar las opciones de un dispositivo Bluetooth.
2. Encontrar otros dispositivos Bluetooth en tu entorno.
3. Conectar a uno de esos dispositivos Bluetooth y comunicarte con ellos.

4. J4ME *[2]*

El framework J4ME ha sido usado en Geomantes como forma de:

- Acceder a los datos de localización de un dispositivo GPS Bluetooth.
- Hacer que el comportamiento del JSR-179 sea consistente.

J4ME " envuelve " y extiende el comportamiento de JSR-179 y hace más sencilla la labor del programador, dejando a este framework las tareas más tediosas. La funcionalidad añadida permite a la aplicación obtener la información de localización de un GPS Bluetooth y no necesariamente integrado. Esto hace a nuestra aplicación una más aplicable ya que, a día de hoy, son muy superiores en número los dispositivos Bluetooth que los que tienen un GPS integrado.

Otra ventaja de J4ME es que hace consistente la implementación del API JSR-179. Algunos dispositivos, pese a ser compatibles con dicho API, provocan excepciones, que gracias a J4ME, son tratadas en un hilo no principal, permitiendo que la ejecución del programa continúe.

Requiere CLDC 1.1/MIDP 2.0. Además, necesita el API Bluetooth (JSR-82) para hacer uso de Bluetooth y/o el API de localización (JSR-179) para obtener los datos GPS. De esta forma, un dispositivo que solo tiene Bluetooth API pero no Location API, puede hacer uso de un GPS Bluetooth.

A parte de lo ya mencionado, J4ME presta otros dos servicios muy destacables (aunque no utilizados en Geomantes): una interfaz de usuario profesional e intuitiva y un sistema de logging para hacer posible una depuración del programa en dispositivos móviles.

5. RMS: Record Management Store *[4]*

El uso de RMS es debido a la necesidad de Geomantes de guardar determinada información incluso una vez ha sido cerrada la aplicación. Información relativa al uso de los dispositivos GPS, de los puntos de interés o de las rutas donde se mantienen almacenados los mapas en el móvil. Toda esa información no puede ser almacenada en un archivo externo al programa por restricciones propias de J2ME. Es por ello por lo que hay que usar RMS: se trata de un sistema de bases de datos que permite añadir información en una memoria no volátil del móvil.

En una base de datos RMS, el elemento básico es el registro (record). Un registro es la unidad de información más pequeña que puede ser almacenada. Los registros son almacenados en un recordStore que puede visualizarse como una colección de registros. Cuando almacenamos un registro en el recordStore, a éste se le asigna un identificador único que identifica unívocamente al registro.

Para poder utilizar RMS hemos de importar el paquete `javax.microedition.rms`.

1. Abrir y cerrar un recordStore: Antes de poder almacenar un registro hemos de abrir un recordStore con el método openRecordStore().

   ```
   static RecordStore openRecordStore(String nombre, bolean crear)
   ```

   El parámetro nombre es el nombre de la base de datos. El nombre puede tener un tamaño de 32 caracteres. El parámetro crear, si tiene su valor a true, creará la base de datos si no existe. Cuando creamos un recordStore, sólo puede ser accedido desde la suite de MIDlets que la creó.

   Cuando terminamos de utilizar el recordStore, hemos de cerrarlo:

   ```
   RecordStore.closeRecordStore();
   ```

2. Añadir registros: Una vez abierto nuestro recordStore podemos comenzar a añadir registros con el método addRecord().

   ```
   public int addRecord(byte[] dato,int offset, int numBytes)
   ```

   El primer parámetro es el dato que queremos almacenar. Es un array de bytes. El offset es la posición a partir de la cual (dentro del array) se va a almacenar el dato. Finalmente, numBytes es el número de bytes que se van a almacenar. El método retorna el identificador que el RMS ha asignado al registro.

   El método addRecord puede lanzar la excepción RecordStoreException, por lo tanto hemos de capturarla.

   ```
   try {
       int id = recordStore.addRecord (datos, 0, datos.length);
   } catch (RecordStoreException e) {}
   ```

3. Leer registros: El método getRecord() permite acceder al registro que deseemos, siempre que conozcamos su identificador.

   ```
   public byte[] getRecord(int Id)
   ```

No es necesario que almacenemos y mantengamos una lista con todos los identificadores de los registros. Un poco más adelante veremos el método recordEnumeration que nos permitirá conocer el identificador de cada registro. Al igual que con el método addRecord(), hemos de capturar la excepción RecordStoreException.

```
byte[] dato = null;

try {
    dato = recordStore.getRecord(id);
} catch (RecordStoreException e) {}
```

4. Borrar registros: El borrado de registros se realiza con el método deleteRecord().

```
public void deleteRecord(int recordId)
```

Al igual que con la escritura y lectura de registros hemos de tener en cuenta que puede provocar la excepción RecorStoreException**.**

```
try {
    recordStore.deleteRecord(id);
} catch (RecordStoreException e) {}
```

6. GOOGLE MAPS

Servicio de imágenes de mapas que Google ofrece de forma gratuita. En funcionamiento a partir del día 8 de febrero de 2005[cita wiki] consta de las siguientes características básicas:

- Posibilidad de acercarse y alejarse a modo de zoom.
- Búsqueda por coordenadas.
- Búsqueda por dirección física o nombre de ciudades (p. ej. “ Leicester Square, London ”) nos devolvería un mapa con esa posición como punto central.
- Busqueda de comercios.
- Planificación de rutas según diferentes criterios (distancia, tiempo…) y medio de transporte (a pie, en coche…).

- Cuatro vistas distintas: mapa, satélite, relieve, y en distancias cortas, fotografías reales.

En el caso del API usado para nuestra aplicación móvil, hacemos uso de las tres primeras opciones.

Entrando en aspectos más técnicos concernientes al uso que hace Geomantes del servicio de Google Maps, se ha diseñado una estructura matricial que mantiene siempre 9 mapas en memoria, de tal forma que la carga de estos sea transparente para el usuario. Se comienza cargando un **mapa central del tamaño exacto de la pantalla del dispositivo móvil** centrado en las coordenadas actuales que se reciben del GPS. El resto de mapas adyacentes, de igual tamaño, son almacenados en nuestra estructura.

Un mapa queda perfectamente identificado por tres elementos:

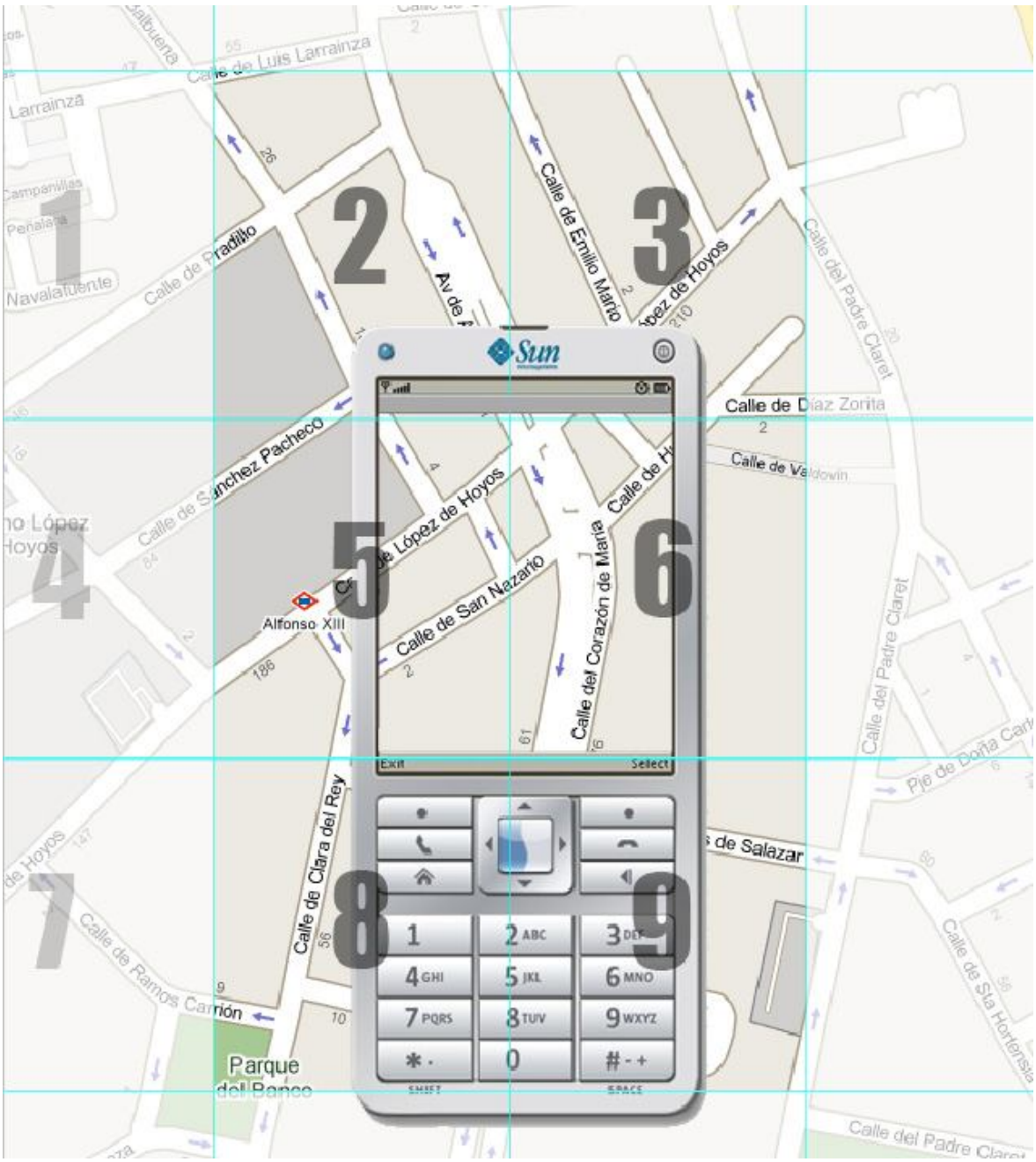
```
public class ElementoGmaps {

    private Image imagen;
    private double longitud;
    private double latitud;
```

A continuación, se detalla cómo van cargándose mapas en la estructura a medida que nos vamos moviendo por los mapas:

Como situación inicial, nos encontramos con nueve mapas cargados y perfectamente centrados en el punto en el que nos encontramos.

Al realizar un desplazamiento lateral, nos movemos hacia la derecha (hacia la izquierda es simétrico, por lo que no se explica) y los tres fragmentos de mapa correspondientes a los números 1, 4 y 7 dejan de sernos útiles.



Finalmente, eliminamos los mapas 1, 4 y 7 de nuestra esctructura y cargamos tres mapas por el lado derecho (simétricamente, izquierdo), de tal forma que tenemos todos los laterales ya cargados por si el usuario se desplaza fuera del mapa central.

★★★★★★★★★★

Se ha implementado una función para poder utilizar los mapas de Google Maps de forma autónoma, es decir, sin conexión a la red. Para ello, Geomantes almacena los mapas que se van cargando en memoria y genera un fichero .map con la información de geolocalización necesaria para el correcto funcionamiento del posicionamiento mediante ruta.

★★★★★★★★★★

7. FILE BROWSING (navegador de archivos)

Para el uso de la aplicación es indispensable acceder a archivos almacenados en el dispositivo. Por ello se hace necesaria la implementación de un navegador de archivos que permita a los usuarios navegar y poder acceder a los distintos archivos y de esa manera, por ejemplo, cargar una ruta deseada o un mapa. Además se le ha añadido la funcionalidad de, en vez de abrir un archivo, guardar la ruta de una carpeta para luego hacer uso de todos los archivos contenidos en esta. Esto es realmente útil para, entre otras cosas, especificar donde se buscan los mapas.

Su compatibilidad con los distintos dispositivos es muy amplia ya que no usa ningún API especial que lo haga dependiente de ninguna tecnología. La única pega es que al acceder al sistema de ficheros del dispositivo en el que corre la aplicación, por motivos de seguridad, son necesarios una gran multitud de permisos y en algunos casos según el fabricante adquirir estos permisos puede ser complicado.

```
boolean isAPIAvailable = false;
if (System.getProperty(
“microedition.io.file.FileConnection.version") != null){
      isAPIAvailable = true;
      System.out.println("hi");
      try {
            new Thread(
                  new Runnable() {
                        public void run(){
                              showCurrDir();
                        }
                  }
            ).start();
      }
      catch (SecurityException e){
            System.out.println(e);
      }
      catch (Exception e) {System.out.println(e);
      }
}
else
{
      String info =
      controlador.getFileConnectionInfo(isAPIAvailable);
      Alert splash = new Alert(null,info,null,AlertType.INFO);
      splash.setTimeout(3000);
      controlador.setDisplay(splash);
}
```

Como se observa del fragmento de código anterior, nuestro navegador de ficheros es un hilo que se crea con tan solo una restricción: que la llamada `System.getProperty( “microedition.io.file.FileConnection.version")` devuelva un valor distinto a `null`.

Como se ha descrito anteriormente, el navegador dispone de dos funcionamientos bien diferenciados:

En la carga de una carpeta para el análisis posterior de su contenido (puntos de interés, archivos .map, contenedor de mapas…), se realizan las siguientes operaciones:

```
else if (c == cargar) {
      List curr = (List) d;
      final String currFile = curr.getString(
      curr.getSelectedIndex());
      System.out.println("currFile:"+currFile);
      ruta = "file:///" + currDirName +currFile;
      fin = true;
}
```

En la carga de un archivo, en cambio, la cantidad de operaciones que intervienen es mucho más compleja, ya que hay que distinguir entre muchos más factores:

```
if (c == view) {
      List curr = (List) d;
      final String currFile = curr.getString(
      curr.getSelectedIndex());
      System.out.println("currFile:"+currFile);

      new Thread(new Runnable() {
            public void run() {
                  if (currFile.endsWith(SEP_STR) ||
                  currFile.equals(UP_DIRECTORY)) {
                        traverseDirectory(currFile);
                  } else {
                        ruta = "file:///" + currDirName +
                        currFile;
                        fin = true;
                  }
            }
       }          ).start();
}
```

Sí `(currFile.endsWith(SEP_STR) || currFile.equals(UP_DIRECTORY))` entonces continuamos por el árbol de la estructura de archivos. En caso contrario, almacenamos la ruta actual y damos por finalizada la búsqueda.

EL GPS Y LA GEOLOCALIZACIÓN

1. DESCRIPCIÓN

El principal reto a la hora de comenzar a planificar Geomantes era conseguir conectar de una forma eficiente y simple nuestra aplicación a un dispositivo GPS. Pese a que esa tarea no implica grandes complicaciones, descubrimos que debíamos adquirir un gran número de conceptos que están asociados al mundo de la geolocalización y la navegación con GPS. Todos estos conocimientos, así como su aplicación en nuestro proyecto, es lo que queda reflejado en las siguientes páginas.

2. EL GPS *[1] [5]*

Las siglas de GPS responden a Sistema de Posicionamiento Global (o Global Positioning System). Gracias a este, podemos determinar la posición de cualquier tipo de objeto con una gran precisión. El sistema GPS está compuesto por tres componentes fundamentales.

- Red de satélites: Cubren toda la superficie de la tierra repartidos en 6 planos con 4 satélites en cada uno (para un total de 24 satélites). Se alimentan gracias a los paneles solares que los constituyen. Tienen una vida útil media de 7 años y medio y se encuentran a una altitud de 20.200 km sobre el nivel del mar.
- Estaciones terrestres: Se intercomunican con los satélites para mantener operativa la estructura.
- Terminales receptores: Más conocidas como Unidades GPS, permiten conocer la posición en la que se encuentran.

La posición del terminal se mide calculando la distancia entre este y los satélites. Esto se hace midiendo el tiempo de respuesta hasta que alcanza el receptor.

La fiabilidad de la señal obtenida por un receptor varía en torno a una serie de parámetros; las nubes, montañas o edificios presentan un problema ineludible que puede reducir la precisión de la señal considerablemente. En condiciones favorables (7, 8 o 9 satélites visibles), se obtienen altísimas precisiones de entorno a 2 metros con una probabilidad superior al 95%. En la siguiente tabla se observan las fuentes de error más comunes y su efecto asociado:

Fuente	Efecto
Ionosfera	± 5 metros
Efemérides	± 2.5 metros
Reloj Satelital	± 2 metros
Distorsión multibanda	± 1 metro
Troposfera	± 0.5 metros
Errores numéricos	± 1 metro (o inferior)

3. LOCALIZACIÓN GEOGRÁFICA Y TIPOS DE COORDENADAS *[16] [17]*

La localización geográfica de un punto se puede realizar detallando uno de estos dos parámetros:

- Coordenadas geográficas en formato Longitud-Latitud.
- Coordenadas UTM (Universal Transverse Mercator).

Asimismo, para poder representar un punto en un mapa con sus valores de coordenadas, es necesario emplear un modelo matemático conocido como *Datum.*

3.1 Coordenadas Geográficas

Las coordenadas geográficas son una forma de designar un punto sobre la superficie terrestre teniendo en cuenta dos medidas: longitud y latitud. El formato es el siguiente:

3º45 ' 25 ' ' W
32º11 ' 44 ' ' N

Estas medidas representan la distancia en grados, minutos y segundos con respecto al meridiano de Greenwich en el caso de la longitud, y con respecto al ecuador en el caso de la latitud.

3.2. Meridianos

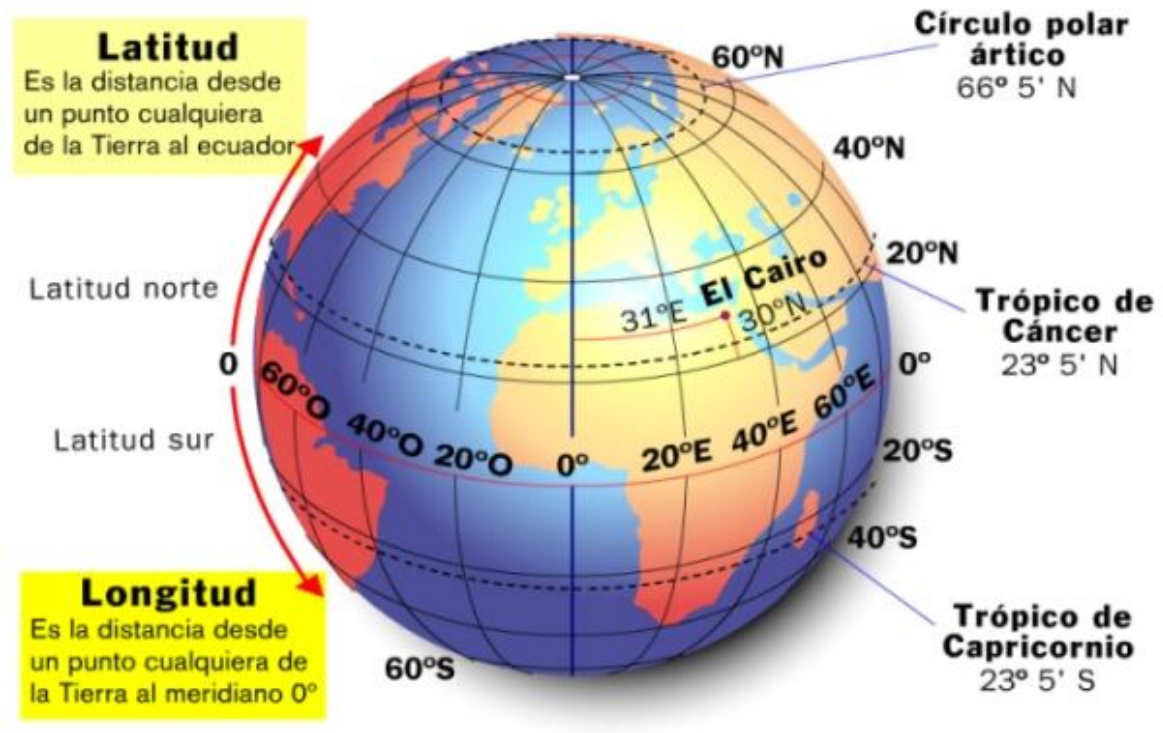
Se definen los meridianos como las líneas de intersección con la superficie terrestre de los infinitos planos que contienen el eje de la Tierra.

La existencia del meridiano 0º o meridiano de Greenwich divide al globo terráqueo en dos zonas: las situadas al oeste (W) del meridiano 0º hasta el antimeridiano, y las situadas al este (E) del meridiano 0º hasta el antimeridiano.

3.3. Paralelos

Se definen los paralelos como las líneas de intersección de los infinitos planos perpendiculares al eje terrestre con la superficie de la Tierra.

Al paralelo de mayor radio se le denomina ecuador. Este paralelo principal divide a la Tierra en dos hemisferios: el norte y el sur. Paralelos geométricamente a él, se trazan el resto de paralelos, de menor radio, tanto en dirección al polo norte como al polo sur.



Representación gráfica de la división de la Tierra en meridianos y paralelos

3.4 Datum [6]

La Tierra no es esférica, ni siquiera es un cuerpo regular achatado por los polos. Esta irregularidad hace que cada país, o incluso cada región, escoja el modelo de cuerpo (definible matemáticamente) que más se ajuste a la forma de la tierra en su territorio. Este cuerpo suele ser un elipsoide.

Los diferentes elipsoides se diferencian unos de otros en sus parámetros, entre los que se encuentran:

- El radio mayor y menor del elipsoide. (a y b).
- El aplastamiento del elipsoide (1/f = 1-(b/a)).

Cada datum está compuesto por:

a) Un elipsoide.
b) Un punto llamado "Fundamental" en el que el elipsoide y la tierra son tangentes. De este punto se han de especificar longitud, latitud y el acimut de una dirección desde él establecida.

En el punto Fundamental, las verticales de elipsoide y tierra coinciden. También coinciden las coordenadas astronómicas (las del elipsoide) y las geodésicas (las de la tierra).

Definido el Datum, ya se puede elaborar la cartografía de cada lugar, pues se tienen unos parámetros de referencia.

Datum en Geomantes

Nuestra aplicación es capaz de interpretar mapas calibrados en distintos Datum, en concreto en los que se presentan a continuación en un fragmento de código fuente:

```
public static Ellipsoid ellipsoid[] = {//  id, nombre del elipsoide,
radio ecuatorial, cuadrado de la excentricidad
        new Ellipsoid(0, "Placeholder", 0.0, 0.0),
        new Ellipsoid(1, "Airy", 6377563, 0.00667054),
        new Ellipsoid(2, "Australian National", 6378160,
0.006694542),
        new Ellipsoid(3, "Bessel 1841", 6377397, 0.006674372),
        new Ellipsoid(4, "Bessel 1841 (Nambia) ", 6377484,
0.006674372),
        new Ellipsoid(5, "Clarke 1866", 6378206, 0.006768658),
        new Ellipsoid(6, "Clarke 1880", 6378249, 0.006803511),
        new Ellipsoid(7, "Everest", 6377276, 0.006637847),
        new Ellipsoid(8, "Fischer 1960 (Mercury) ",
6378166,0.006693422),
        new Ellipsoid(9, "Fischer 1968", 6378150, 0.006693422),
        new Ellipsoid(10, "GRS 1967", 6378160, 0.006694605),
        new Ellipsoid(11, "GRS 1980", 6378137, 0.00669438),
        new Ellipsoid(12, "Helmert 1906", 6378200, 0.006693422),
        new Ellipsoid(13, "Hough", 6378270, 0.00672267),
        new Ellipsoid(14, "International", 6378388, 0.00672267),
        new Ellipsoid(15, "Krassovsky", 6378245, 0.006693422),
        new Ellipsoid(16, "Modified Airy", 6377340, 0.00667054),
        new Ellipsoid(17, "Modified Everest", 6377304, 0.006637847),
        new Ellipsoid(18, "Modified Fischer 1960", 6378155,
0.006693422),
        new Ellipsoid(19, "South American 1969", 6378160,
0.006694542),
        new Ellipsoid(20, "WGS 60", 6378165, 0.006693422),
        new Ellipsoid(21, "WGS 66", 6378145, 0.006694542),
        new Ellipsoid(22, "WGS 72", 6378135, 0.006694318),
        new Ellipsoid(23, "WGS 84", 6378137, 0.00669438),
        new Ellipsoid(24, "European 1950", 6378388, 0.00672267)
    };
```

Como se puede observar, Geomantes proporciona una alta flexibilidad a la hora de trabajar con distintos modelos matemáticos de georreferenciación.

Sistema WGS-84

Es el sistema de referencia terrestre adoptado por el Departamento de Defensa de los Estados Unidos para el posicionamiento GPS. WGS-84 es un sistema de coordenadas geocéntrico global basado originariamente en observaciones Doppler del sistema de satélites TRANSIT.

El elipsoide de referencia de WGS-84 es esencialmente el del Sistema Geodésico de referencia 1980 (GRS 80), de la Unión Geodésica y Geofísica Internacional, con cambios menores, sólo en su aplastamiento. Los parámetros del elipsoide WGS-84 son:

Semieje mayor a= 6.378.137 m. y Aplastamiento f = 1/ 298,257223563

A día de hoy, la gran mayoría de los receptores GPS generan internamente los datos en el sistema WGS-84.

3.5. Proyecciones

La proyección cartográfica permite representar una superficie esférica como la Tierra en una lámina de papel plana. Una proyección cartográfica es una representación sistemática de los paralelos y meridianos de una superficie tridimensional en una superficie bidimensional. Dado que una superficie plana no puede ajustarse a una esfera sin estirarse o encogerse tampoco es posible representar atributos de un globo en un mapa sin causar distorsiones.

Existen diversas proyecciones y cada una de ellas trata de minimizar las distorsiones. Las proyecciones que se utilizan en la actualidad se han derivado a partir de modelos matemáticos del globo terrestre y todas ellas comparten la misma característica: mostrar la posición correcta de las líneas de longitud y latitud del Planeta. En otras palabras, cada proyección es solamente un reordenamiento de los meridianos y paralelos traslados del Globo Terrestre a un mapa. Dado que no hay forma de eliminar los errores al trasladar una superficie curva (Tierra) a una superficie plana (mapa) ninguna proyección es geométricamente perfecta. En síntesis, cada proyección es elaborada a partir de una figura geométrica con un propósito particular y por ende tiene sus propias virtudes y limitaciones.

Proyección UTM (Universal Transverse Mercator)

Para calcular distancias de un punto geográfico a otro dadas sus longitudes y latitudes, debemos llevar a cabo una conversión a coordenadas UTM. La proyección UTM (Universal Transverse Mercator) tiene su origen en el desarrollo cilíndrico conforme de Gauss.

La superficie de referencia utilizada es un elipsoide de revolución al cual se le situará tangente exteriormente un cilindro transverso. La proyección se divide en 60 husos de 6º de longitud que se empiezan a numerar desde el antimeridiano origen, de forma que el meridiano de Greenwich define el límite entre los husos 30 (al oeste) y 31 (al este). España está, de este modo, comprendida entre los husos 28 al 31. El origen de coordenadas Y es el ecuador mientras que las coordenadas X se calculan desde el meridiano central, al que se le asigna un valor de 500000 metros con el fin de evitar coordenadas negativas.

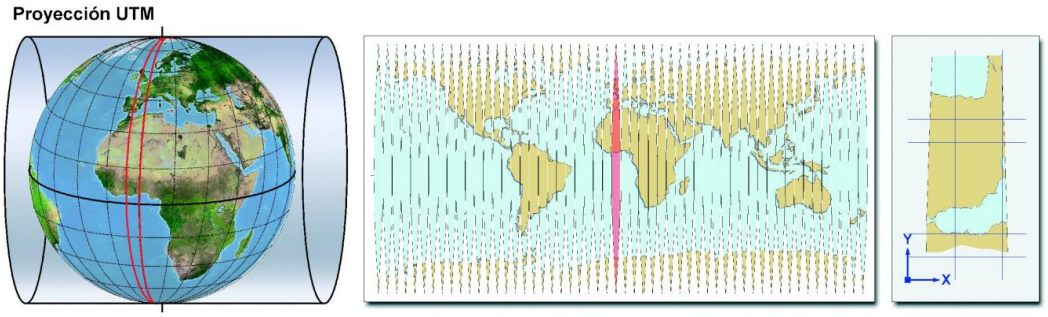


Diagrama explicativo de la proyección UTM

4. OZIEXPLORER Y LOS FICHEROS .MAP *[1] [15]*

La geolocalización de los mapas cartográficos solo es posible si asociamos a las imágenes un fichero con la información necesaria. Nuestra elección por motivos de compatibilidad ha sido la de usar el estándar de OziExplorer.

OziExplorer es un programa dedicado al mundo de GPS. Tiene muchas ventajas. Una de ellas es que es el único programa que hoy día te permite trabajar con los GPS de los modelos Garmin, Lowrance, y Magellan, permitiendo el intercambio de datos (rutas, tracks y waypoints) entre la práctica totalidad de los usuarios de GPS. Otros aspectos a destacar es su versatilidad (usa mapas escaneados o digitales), importa mapas, crea rejillas (grids), tiene implementada la función de mapa móvil y la de auto-pilotaje, y un largo etcétera. Además, es extraordinariamente preciso, y sobre todo, posee el mejor apoyo técnico de todos los programas. Su autor, Des Newman, contestará a todas tus preguntas, e incluirá tus sugerencias.

Dicho programa utiliza la extensión " .map " para identificar a los archivos de geolocalización. A continuación se expone un ejemplo y se detallan algunas de sus características:

```
OziExplorer Map Data File Version 2.2
CasaDoria.jpg
CasaDoria.jpg
1 ,Map Code,
WGS 84,, 0.0000, 0.0000,WGS 84
Reserved 1
Reserved 2
Magnetic Variation,,,E
Map Projection,(UTM) Universal Transverse
Mercator,PolyCal,No,AutoCalOnly,No,BSBUseWPX,No
Point01,xy,0,0,in, deg, , ,N, , ,W, grid, 30, 442259.0041454903,
4479280.670948872,N
Point02,xy,2048,0,in, deg, , ,N, , ,W, grid, 30, 444307.0006342536,
4479280.654703642,N
Point03,xy,2048,2048,in, deg, , ,N, , ,W, grid, 30, 444306.98426401743,
4477232.658259679,N
Point04,xy,0,2048,in, deg, , ,N, , ,W, grid, 30, 442258.9877889765,
4477232.674504637,N
Projection Setup,,,,,,,,,,
Map Feature = MF ; Map Comment = MC     These follow if they exist
```

Sistema de coordenadas utilizado (en este caso, WGS 84).

```
Track File = TF      These follow if they exist
Moving Map Parameters = MM?    These follow if they exist
MM0,Yes
MMPNUM,4
MMPXY,1,0,0
MMPXY,2,2048,0
MMPXY,3,2048,2048
MMPXY,4,0,2048
MMPLL,1,-3.6810547207149833,40.46219122467416
MMPLL,2,-3.6569001464900044,40.46233087535915
MMPLL,3,-3.656720698347238,40.44388149386048
MMPLL,4,-3.6808686681269895,40.443741933683825
MM1B,1.0
MOP,Map Open Position,0,0
IWH,Map Image Width/Height,2048,2048
```

Información de las cuatro esquinas que delimitan el

Coordenadas asociadas a las esquinas del mapa

5. GLOSARIO DE TERMINOLOGÍA

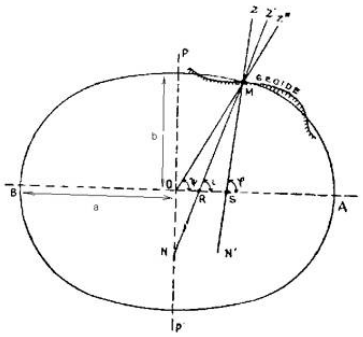


Fig. 8

Datum: Dadas las irregularidades de la superficie de la tierra, cada país ha de escoger un modelo que se ajusta a la forma de su territorio. Normalmente se suele tratar de un elipsoide. Cada datum se compone por este elipsoide y un punto denominado " Fundamental ", en el que la tierra y el elipsoide son tangentes. De esta forma, definido el datum, tenemos los datos necesarios para elaborar la cartografía de un lugar.

WGS84: Son las siglas de World Geodetic System 1984. Sistema de coordenadas mundiales, que data de 1984, que es la base para sistemas de posicionamiento globales como el GPS. Está pendiente una próxima revisión para 2010.

El WGS84 usa como elipsoide de referencia a WGS 84, que es perfectamente intercambiable con el sistema de referencia europeo ETRS 89, estribando su única diferencia en varias décimas de milímetro para el caso de los semiejes menores. *[18]*

ED50: Para el oeste europeo encontramos un tipo de datum denominado ED50 (European Datum 1950) de después de la II Guerra Mundial. Surge por la necesidad de unificar los datos de mapas de países como Alemania, Holanda, Bélgica o Francia, que tenían posicionamientos de latitud y longitud incompatibles. Este sistema tiene su centro al sur de Alemania, donde se situó el centro de la Europa occidental a finales de la guerra fría.

Raster: cartografía en imagen o RASTER es la navegación a través de imágenes cartográficas (u ortomapas) indicándonos la posición exacta que marca la señal GPS al contrario de sistemas como TomTom, que asignan nuestra posición al camino más cercano mediante una especie de " imantación " .

Huso: Geográficamente los husos horarios son cada una de las veinticuatro áreas en que se divide la Tierra. Se llaman así porque tienen forma de huso de hilar o de gajo de naranja y están centrados en meridianos de una longitud que es múltiplo de 15°.

DESARROLLO DEL SISTEMA

1. Casos de uso *[7] [8]*

Un modelo de caso de uso es una especificación de funcionalidad describiendo la interacción del sistema con los actores a veces apoyada en una representación gráfica para facilitar la comprensión pero construida principalmente a partir de texto. Consta normalmente de las siguientes partes:

- Código: Abreviatura que simplifica las referencias posteriores al CU.
- Nombre: Identificador comprensible.
- Actores: Genéricamente se trata de un representante de un papel en un guión. En nuestro caso usamos esa metáfora para describir a un agente externo que interactúa con el sistema. Pese a lo anterior, ciertas cosas generalmente vistas como externas al sistema no son considerados buenos actores. Los sistemas de persistencia como las bases de datos o los archivos, si bien externos, no son actores.
- Descripción: Breve explicación acerca de lo que ocurre durante el CU.
- Precondición: Eventos que han de cumplirse antes del CU.
- Postcondición: Acciones que se cumplen una vez finalizado el CU.
- Flujo principal: Orden de acciones que se completan durante la CU en una ejecución sin imprevistos. Solo se incluye en caso de contener información relevante.
- Flujo alternativo: Posibles desvíos sobre el flujo principal. Solo se incluye en caso de contener información relevante.

CU-0101	**Navegación libre en GM**
Actores	Usuario.
Descripción	Acceso a la navegación mediante GPS con mapas del servicio de Google Maps.

Precondición	Disponer de un dispositivo GPS y conexión a internet.
Postcondición	Se permite la libre navegación.
Flujo principal	1. Seleccionar Mapas. 2. Seleccionar Modo Google Maps. 3. Seleccionar Navegación. 4. Seleccionar Navegación libre.
Flujo alternativo	1.1. Si el dispositivo GPS no funciona, se aborta la secuencia. 2.1. Si no se dispone de conexión a internet, se aborta la secuencia.

CU-0102	**Navegación libre en mapas propios**
Actores	Usuario.
Descripción	Acceso a la navegación mediante GPS con mapas almacenados en el móvil.
Precondición	Disponer de algún mapa previamente almacenado en el dispositivo móvil, además de acceso a un dispositivo GPS.
Postcondición	Se permite la libre navegación.
Flujo principal	1. Seleccionar Mapas. 2. Seleccionar Cargar Mapa. 3. Navegar por el sistema de archivos hasta encontrar el mapa que deseamos cargar. 4. Seleccionar un mapa para cargar. 5. Seleccionar Navegación. 6. Seleccionar Navegación libre.
Flujo alternativo	3.1. Si decidimos retroceder, volveremos a la pantalla

	principal. 4.1. Si el mapa cargado no tiene un formato reconocido, se vuelve a la pantalla principal.

CU-0103	**Navegación guiada en GM**
Actores	Usuario.
Descripción	Acceso a la navegación mediante GPS con mapas almacenados en el móvil siguiendo una ruta previamente cargada.
Precondición	Disponer de dispositivo GPS, conexión a internet y una ruta previa.
Postcondición	Se permite la navegación guiada a través del servicio Google Maps.
Flujo principal	1. Seleccionar Mapas. 2. Seleccionar Modo Google Maps. 3. Seleccionar Rutas. 4. Seleccionar Cargar Ruta. 5. Navegar por el sistema de archivos hasta encontrar la ruta que deseamos cargar. 6. Seleccionar una ruta para cargar. 7. Seleccionar Navegar. 8. Seleccionar Navegación guiada.
Flujo alternativo	5.1. Si decidimos retroceder, volveremos a la pantalla principal. 6.1. Si la ruta cargada no tiene un formato reconocido, se vuelve a la pantalla principal.

CU-0104	Navegación guiada en mapas propios
Actores	Usuario.
Descripción	Acceso a la navegación mediante GPS, siguiendo una ruta, con mapas almacenados en el móvil.
Precondición	Disponer de algún mapa previamente almacenado en el dispositivo móvil, conexión GPS y alguna ruta disponible.
Postcondición	Se permite la navegación guiada sobre mapas propios.
Flujo principal	1. Seleccionar Mapas. 2. Seleccionar Cargar Mapa. 3. Navegar por el sistema de archivos hasta encontrar el mapa que deseamos cargar. 4. Seleccionar el mapa para cargar. 5. Seleccionar Rutas. 6. Seleccionar Cargar Ruta. 7. Navegar por el sistema de archivos hasta encontrar la ruta que deseamos cargar. 8. Seleccionar una ruta para cargar. 9. Seleccionar Navegar. 10. Seleccionar Navegación guiada.
Flujo alternativo	2.1. Si decidimos retroceder, volveremos a la pantalla principal. 4.1. Si el mapa cargado no es reconocido, se vuelve a la pantalla principal.

	7.1. Si decidimos retroceder, volveremos a la pantalla principal. 8.1. Si la ruta cargada no es reconocida, se vuelve a la pantalla principal.

CU-0105	**Reproducir ruta en GM**
Actores	Usuario.
Descripción	Recreación de una ruta ya recorrida mediante mapas de Google Maps.
Precondición	Disponer de conexión a internet.
Flujo principal	1. Seleccionar Mapas. 2. Seleccionar Modo Google Maps. 3. Seleccionar Rutas. 4. Seleccionar Cargar Ruta. 5. Navegar por el sistema de archivos hasta encontrar la ruta que estamos buscando. 6. Seleccionar la ruta deseada. 7. Seleccionar Ver demo rutas. 8. Seleccionar modo Google Maps.
Flujo alternativo	8.1. Si la conexión a internet falla, se vuelve a la pantalla principal.

CU-0106	**Reproducir ruta en mapa propio**
Actores	Usuario.
Descripción	Recreación de una ruta ya recorrida mediante mapas almacenados en el móvil.
Precondición	Disponer de mapas previamente

	cargados en el dispositivo móvil.
Flujo principal	1. Seleccionar Modo Offline. 2. Seleccionar Ver demo rutas. 3. Seleccionar modo Cargar Mapa. 4. Navegar por el sistema de archivos hasta encontrar el mapa deseado. 5. Cargar la ruta escogida. 6. Seleccionar Reproducir ruta.
Flujo alternativo	4.1. Si decidimos retroceder, se vuelve a la pantalla principal. 5.1. Si el mapa escogido no es reconocido por el sistema, se vuelve a la pantalla principal.

CU-0107	**Calibrar mapa desde archivo**
Actores	Usuario.
Descripción	Geolocalización de un fichero que contiene un mapa.
Precondición	Disponer de un dispositivo GPS y algún mapa almacenado en el dispositivo móvil.
Flujo principal	1. Seleccionar Opciones. 2. Seleccionar Mapas. 3. Seleccionar Calibrar mapa. 4. Navegar por el sistema de archivos para encontrar una imagen de mapa. 5. Seleccionar mapa a cargar.
Flujo alternativo	4.1. Si decidimos retroceder, se vuelve a la pantalla principal. 5.1. Si el mapa escogido no es

	reconocido por el sistema, se vuelve a la pantalla principal.

CU-0108	**Ajuste mapa**
Actores	Usuario.
Descripción	Ajustes de propiedades asociadas a los mapas.
Precondición	
Flujo principal	1. Seleccionar Opciones. 2. Seleccionar Mapas. 3. Seleccionar Ajustes mapa.
Flujo alternativo	

CU-0109	**Añadir puntos de interés**
Actores	Usuario.
Descripción	Añadir puntos de interés con las coordenadas que marca el GPS.
Precondición	Disponer de un dispositivo GPS y memoria suficiente en el dispositivo móvil.
Flujo principal	1. Seleccionar Opciones. 2. Seleccionar Puntos de interés. 3. Seleccionar guardar nuevo

CU-0110	**Borrar puntos de interés**
Actores	Usuario.
Descripción	Borrar puntos de interés propios.
Precondición	Disponer de algún punto de interés.
Flujo principal	1. Seleccionar Opciones.

	2. Seleccionar Puntos de interés. 3. Seleccionar borrar puntos interés. 4. Navegar por el sistema de puntos de interés. 5. Seleccionar tantos puntos como se quiere borrar.
Flujo alternativo	4.1 Si decidimos retroceder, se vuelve a la pantalla principal. Si se borran todos los puntos de interés volver a menú principal.

CU-0111	**Cargar archivos puntos de interés.**
Actores	Usuario.
Descripción	Carga archivos de puntos de interés.
Precondición	Disponer de algún archivo de puntos de interés.
Flujo principal	1. Seleccionar Opciones. 2. Seleccionar Puntos de interés. 3. Seleccionar cargar puntos interés. 4. Navegar por el sistema de archivos para encontrar un archivo de puntos de interés. 5. Cargar archivo de puntos de interés.
Flujo alternativo	4.1. Si decidimos retroceder, se vuelve a la pantalla principal. 5.1 Si el archivo no es reconocido por el sistema, se vuelve a la pantalla principal.

CU-0112	Seleccionar idioma
Actores	Usuario.
Descripción	Seleccionar el idioma de la aplicación.
Precondición	
Flujo principal	1. Seleccionar Opciones. 2. Seleccionar Idiomas. 3. Seleccionar un idioma de la lista.
Flujo alternativo	

CU-0113	Grabar ruta
Actores	Usuario.
Descripción	Grabar una ruta y guardarla.
Precondición	Disponer de dispositivo GPS y un recorrido en ejecución (CU-100 a CU-104).
Flujo principal	1. Seleccionar Rutas. 2. Seleccionar Iniciar Grabación. 3. Moverse por la ruta. 4. Finalizar grabación. 5. Dar una dirección para guardar la ruta.
Flujo alternativo	5.1 Si se aborta la operación, se vuelve al menú principal.

CU-0114	Descargar una ruta
Actores	Usuario.
Descripción	Quita de memoria una ruta cargada en memoria.
Precondición	Debe haber una ruta cargada en

	memoria.
Flujo principal	1. Seleccionar Rutas. 2. Seleccionar Descargar ruta.

CU-0115	**Borrar una ruta**
Actores	Usuario.
Descripción	Borra una ruta que esté almacenada en el dispositivo móvil.
Precondición	Debe haber alguna ruta almacenada en el dispositivo móvil.
Flujo principal	1. Seleccionar Rutas. 2. Seleccionar Borrar ruta. 3. Navegar por el sistema de archivos para encontrar un archivo de ruta. 4. Seleccionar la ruta a borrar.
Flujo alternativo	3.1. Si decidimos retroceder, se vuelve a la pantalla principal. 4.1. Si el archivo no es reconocido por el sistema, se vuelve a la pantalla principal.

CU-0116	**Descargar mapa**
Actores	Usuario.
Descripción	Quita de memoria el mapa que este cargado actualmente.
Precondición	Debe haber un mapa cargado en memoria.
Flujo principal	1. Seleccionar Mapas. 2. Seleccionar Descargar mapa.

CU-0117	Borrar mapa
Actores	Usuario.
Descripción	Borra un mapa almacenado en el dispositivo móvil.
Precondición	Debe haber algún mapa almacenado en el dispositivo móvil.
Flujo principal	1. Seleccionar Mapas. 2. Seleccionar Borrar mapa. 3. Navegar por el sistema de archivos para encontrar un mapa. 4. Seleccionar el mapa a borrar.
Flujo alternativo	3.1. Si decidimos retroceder, se vuelve a la pantalla principal. 4.1 Si el archivo no es reconocido por el sistema, se vuelve a la pantalla principal.

CU-0118	Previsualizar mapa
Actores	Usuario.
Descripción	Carga el mapa en pantalla y permite su exploración mediante los cursores.
Precondición	Debe haber algún mapa almacenado en el dispositivo móvil.
Flujo principal	1. Seleccionar Mapas. 2. Seleccionar Previsualizar mapa. 3. Navegar por el sistema de archivos para encontrar un mapa. 4. Seleccionar el mapa a previsualizar 5. Moverse a través de los cursores por el mapa.
Flujo alternativo	3.1 Si decidimos retroceder, se vuelve a la pantalla principal.

	4.1. Si el archivo no es reconocido por el sistema, se vuelve a la pantalla principal. 5.1. Si decidimos cancelar, volvemos a la pantalla principal.

CU-0119	**Opciones GPS**
Actores	Usuario.
Descripción	Se modifican las opciones relacionadas con el dispositivo GPS.
Precondición	
Flujo principal	1. Seleccionar Opciones. 2. Seleccionar GPS.
Flujo alternativo	

CU-0120	**Resto de opciones**
Actores	Usuario.
Descripción	Permite modificar el resto de opciones no englobadas en ningún otro sitio.
Precondición	
Flujo principal	1. Seleccionar Opciones. 2. Seleccionar Otras opciones.
Flujo alternativo	

CU-0121	**Reproducir ruta**
Actores	Usuario.
Descripción	Permite modificar el resto de opciones no englobadas en ningún

	otro sitio.
Precondición	
Flujo principal	1. Seleccionar Opciones. 2. Seleccionar Otras opciones.
Flujo alternativo	

2. Viabilidad y alcance

Tanto para la ***viabilidad*** como para el ***alcance*** de la aplicación lo primero que tenemos que tener en cuenta es '*que es lo que queríamos hacer*' cuando todo esto comenzó, es decir, dejar bien claro cuáles eran los *requisitos funcionales* o servicios que la aplicación debe proporcionar al usuario así como las condiciones de limitación que el sistema podría tener y que influyera en el usuario final que es quien nos tiene que importar, es decir *requisitos no funcionales*.

Pues bien, nuestra meta al principio era construir una aplicación que proporcionase al usuario un sistema fiable de localización mediante GPS. Además, queríamos dar soporte al uso de rutas y distintos tipos de mapas.

A la hora de pensar en que es lo que estamos dispuestos a hacer y como lo haremos para que sea más ventajoso para nuestros usuarios nos preguntamos:

1. ¿Nuestra aplicación contempla todas las funciones que el cliente debería de esperar de ella?
2. ¿Pueden los Requisitos ser implementados con la tecnología y el presupuesto disponible?
3. ¿Tenemos experiencia suficiente como para realizar la planificación y elaboración de nuestros objetivos en el tiempo marcado?

A la *primera pregunta*, en nuestra opinión, tenemos que contestar que sí. Se ofrecen posibilidades más que básicas para un usuario medio, y además se han implementado una serie de opciones muy interesantes para el usuario avanzado.

La *segunda pregunta*, en cuanto al presupuesto se refiere, tenemos que aclarar que aunque era reducido, no contemplaba los salarios de los empleados encargados de desarrollar el proyecto, ya que éramos tres estudiantes, con lo que en presupuesto nos ahorrábamos lo que mayor cuantía suele suponer. En cuanto a las tecnologías e infraestructuras hemos utilizado lo mejor posible los recursos de los que disponíamos, utilizando tecnologías ya conocidas en la mayoría de los casos y utilizando como infraestructuras las que la universidad nos ofrecía (los laboratorios) además de las que nosotros disponíamos.

En cuanto a la *tercer pregunta*, hay que decir que no es que dispusieran de mucha experiencia en planificar proyectos, pero desde un principio definimos bien claro cuáles eran los objetivos que queríamos alcanzar con la aplicación fijando las metas a realizar y, aunque ha habido momentos mejores y momentos peores, globalmente nunca se ha dudado de que alcanzaríamos esas metas.

El **alcance** define toda la funcionalidad de los requisitos que hemos logrado completar de cara a esta entrega final del proyecto. Desde un principio, hemos intentado incluir en el alcance los pilares del proyecto completos y con toda su funcionalidad.

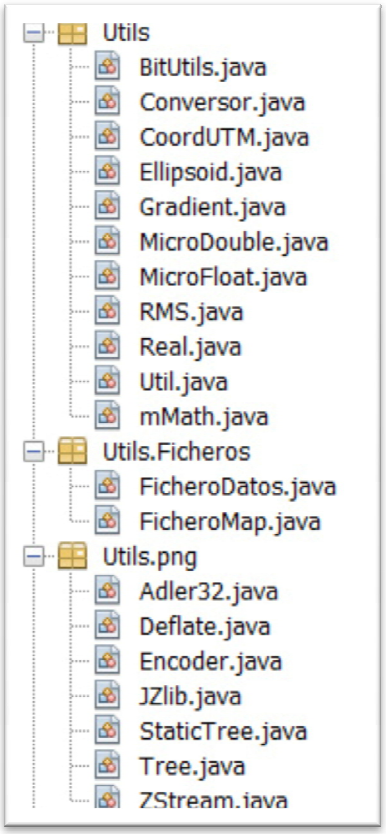
Si bien es verdad que en este tipo de proyectos normalmente hay unas metas principales a lograr que son las que se han alcanzado y que suelen ser el motivo principal del proyecto. También hay algunas funcionalidades que se pretendían alcanzar y no se han alcanzado, bien por motivos de tiempo o por prioridad, y otras que no se pretendían al principio pero luego se han ido planteando y se han logrado realizar. Como ejemplo de lo

primero, el uso de mapas vectoriales, y como ejemplo de lo segundo, la integración con Google Maps.

3. Diseño

El diseño de la aplicación se ha visto determinado desde un principio por los patrones de diseño empleados. En concreto, el patrón de Modelo-Vista-Controlador ha influido directamente en la estructura de clases y paquetes. Se ha tratado de aglutinar las clases semejantes en paquetes que las contuviesen con cierto sentido.

Aparte de los tres paquetes principales (modelo, vista y controlador), se han incluido una serie de paquetes y subpaquetes con utilidades de distinta índole (escritura de ficheros, métodos genéricos de geolocalización, conversores de coordenadas...) denominado **Utils**.



Contenido del paquete *Utils*

3.1 Modelo-Vista-Controlador *[9] [10] [11]*

Modelo Vista Controlador (MVC) es un patrón que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres distintos componentes.

- **Modelo**: Es la representación específica de la información con la que el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comparar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.

- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador**: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

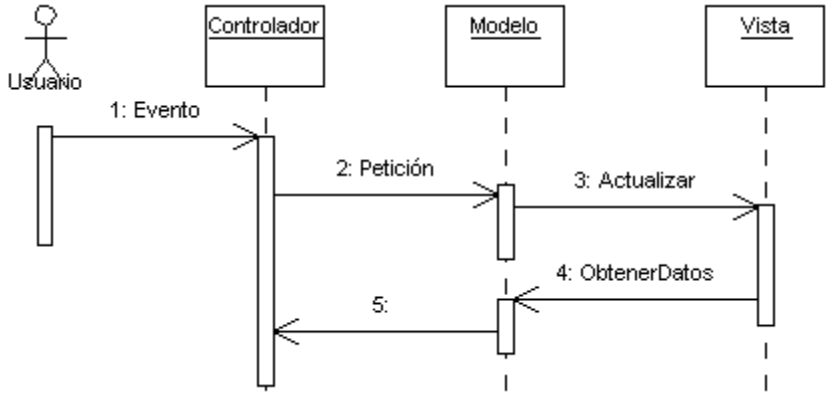


Diagrama que ejemplifica la interactuación entre las tres partes

Un problema muy común para los programadores es la reutilización del código que ya tienen hecho. A veces hay que resolver un problema parecido a algo que ya tenemos hecho, mejorar el aspecto de un programa,

★★★★★★★★★★

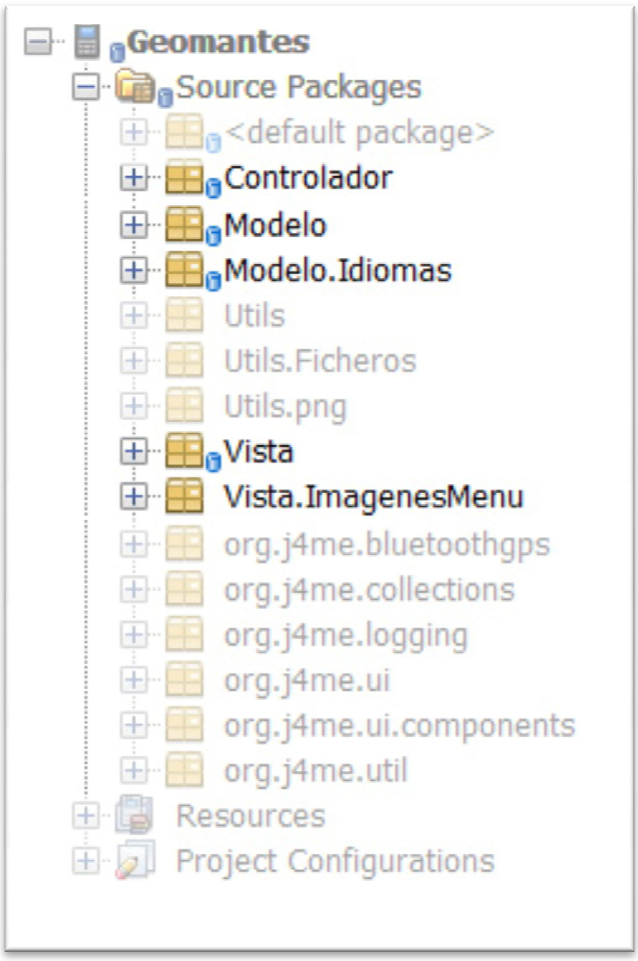
MVC facilita el entendimiento del código potenciando la modularidad y extensibilidad de este.

★★★★★★★★★★

mejorar su algoritmo, etc. Esta tarea se facilita mucho si a la hora de programar tenemos la precaución de separar el código en varias partes que sean susceptibles de ser reutilizadas sin modificaciones.

Lo más reutilizable y que es menos susceptible de cambio, es el modelo. Las fórmulas de geolocalización o el cálculo de distancias no cambian de un día para otro. Si tenemos un conjunto de clases que mantengan en memoria el esta información, es posible que esas clases (o funciones y estructuras de datos) nos sirvan durante mucho tiempo sin necesidad de tocarlas. En un

punto intermedio está el controlador. Finalmente, lo que más cambia, es la vista. De hecho, en un programa como Geomantes es frecuente que se tengan que desarrollar varias presentaciones para según qué información. El modelo y el controlador serían los mismos, pero habría varias vistas distintas.



En la página siguiente se incluye un diagrama que ayuda a entender mejor como Geomantes hace uso del MVC. El gráfico representa el proceso de puesta en funcionamiento de la aplicación a través del tiempo, desde el momento en el que se inicia la carga hasta que comienza a ejecutarse la navegación.

Como se observa a simple vista, Modelo y Vista nunca intercambian información directamente, sino que lo hacen a través del controlador. Esto resalta la modularidad y extensibilidad de la que antes hablábamos.

Lo primero en ser analizado son las preferencias de usuario (*settings*), que determinarán, entre otras cosas, el lenguaje en el que son mostrados los avisos al usuario. A continuación se cargan las vistas, tanto los menús de opciones como la pantalla de navegación. Llegado este punto, la aplicación se encuentra lista para funcionar y responder a lo que el usuario le pida. El control recae sobre la vista, y el controlador pasa a un segundo plano hasta que el usuario ejecuta alguna acción. En ese momento, la vista se lo

comunica al controlador, y este a su vez realiza las gestiones pertinentes tanto con el modelo como con la propia vista.

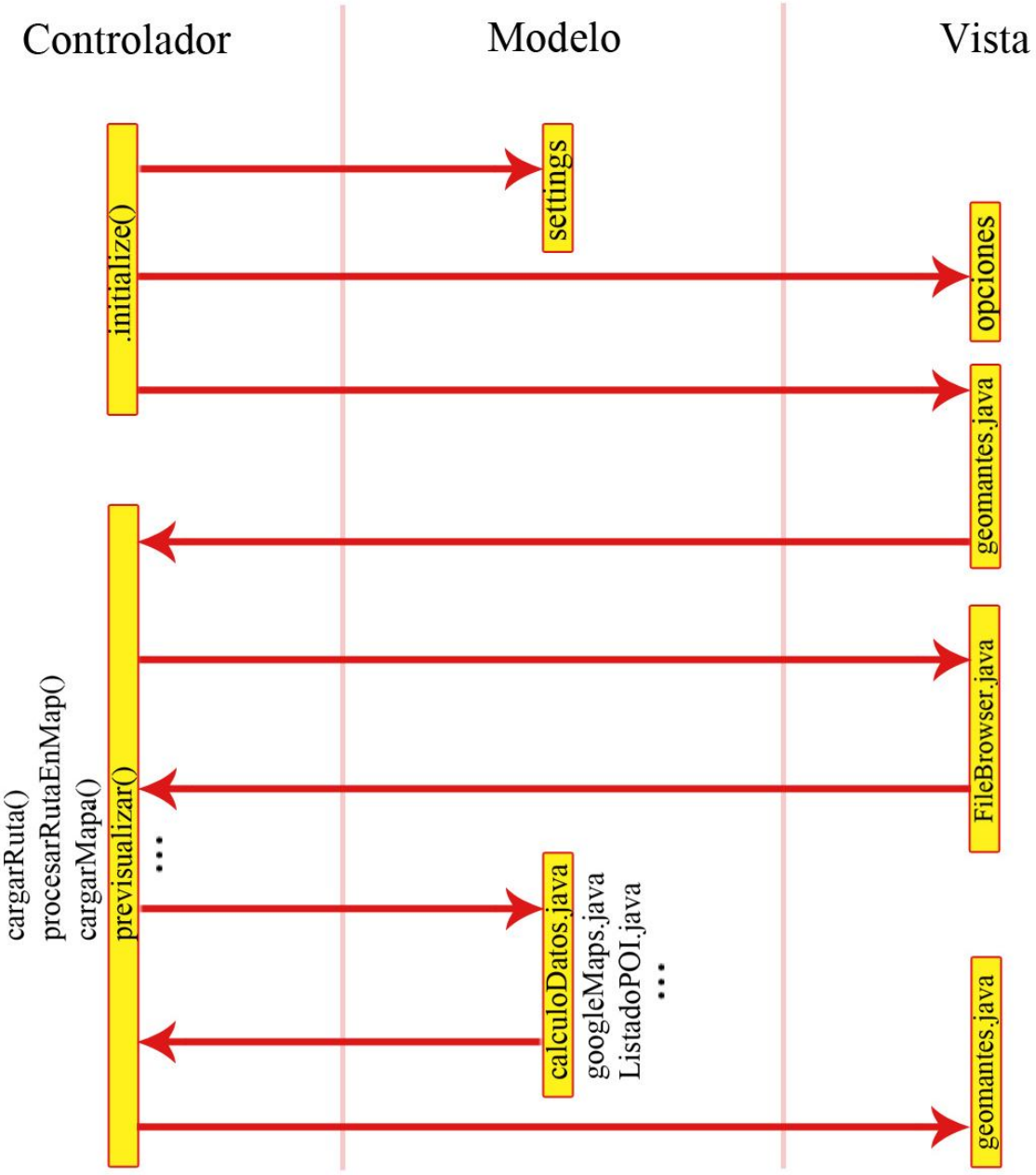
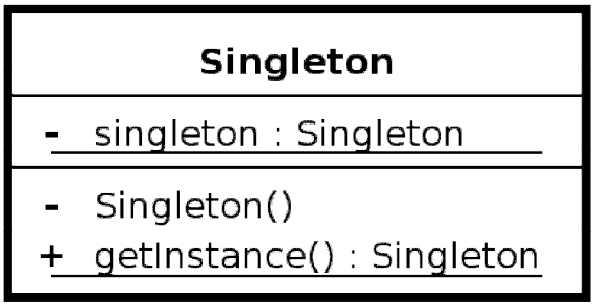


Diagrama explicativo del proceso de carga

Como se observa, una vez que la ejecución se estabiliza de nuevo, o lo que es lo mismo, se le da al usuario lo que ha solicitado, el control pasa de nuevo a la vista. El segundo proceso descrito, se repite tantas veces como sea necesario hasta que se finalice la ejecución.

3.2 Singleton [12] [13] [14]

A lo largo del desarrollo de la aplicación hemos utilizado el patrón Singleton en multitud de ocasiones. Geomantes es una aplicación donde lo que conviene es que no haya más de una instancia de la mayoría de los objetos. Tan solo una pantalla principal, tan solo una ruta activa, un único algoritmo de conversión de coordenadas en activo... El Singleton es el patrón adecuado dadas estas características al estar definido por la creación de instancias únicas.



Se observan dos características básicas:

1. **Restricción de acceso al constructor**: Consigue que sea imposible crear nuevas instancias. Solo la propia clase puede crearla.
2. **Mecanismo de acceso a la instancia**: El acceso a la instancia se hace a través de un único punto. Puede ser alcanzado desde distintos puntos del código, pero siempre está gestionado por la propia clase.

Estructura del patrón utilizado en Geomantes:

```
public class Singleton {
   private static final Singleton INSTANCIA = new Singleton();

   // Constructor privado para prevenir la instanciación externa
   private Singleton() {}

   public static Singleton getInstance() {
      return INSTANCIA;
   }
}
```

PRUEBAS

El periodo de pruebas ha estado presente durante todo el desarrollo de la aplicación, pero ha sido más intenso durante las últimas semanas. Durante este tiempo se han ido detectando problemas que anteriormente habían pasado desapercibidos, de manera que la forma y el contenido de Geomantes se ha ido depurando hasta su versión final.

La **gestión de calidad en el software** tiene como objetivo lograr un nivel de calidad preestablecido en el mismo producto. Se ha de asumir que el precio de la calidad es una gran inversión en tiempo de desarrollo para asegurarse de que al final se obtendrá un producto mejor. Como indicábamos al principio, no puede esperarse a que el producto esté terminado para preocuparse por su calidad. La labor de conseguir calidad ha de estar presente en la definición del proceso de desarrollo.

★★★★★★★★★★

- *Un software es de calidad si se ajusta al propósito para el que se creó (si satisface sus propias especificaciones).*

- *Existen atributos observables que aportan calidad a un software de manera independiente de si satisface o no sus especificaciones.*

    - *Seguro, fácil de usar, modular…*

★★★★★★★★★★

Hemos de tomar definiciones de calidad como pueden ser la ISO 9000 o la ISO 9001, establecer procedimientos organizacionales para la calidad y garantizar que se siguen estos estándares durante el desarrollo del software.

¿En qué se traduce la instauración de mecanismos orientados a la calidad en el desarrollo del software?

1. Definir procesos y estándares que indiquen como llevar a cabo las revisiones de los resultados.
2. Imponer un seguimiento del proceso de desarrollo para asegurar que se están siguiendo estos estándares.
3. Realizar informes sobre los resultados del seguimiento de estos procesos a la administración del proyecto.

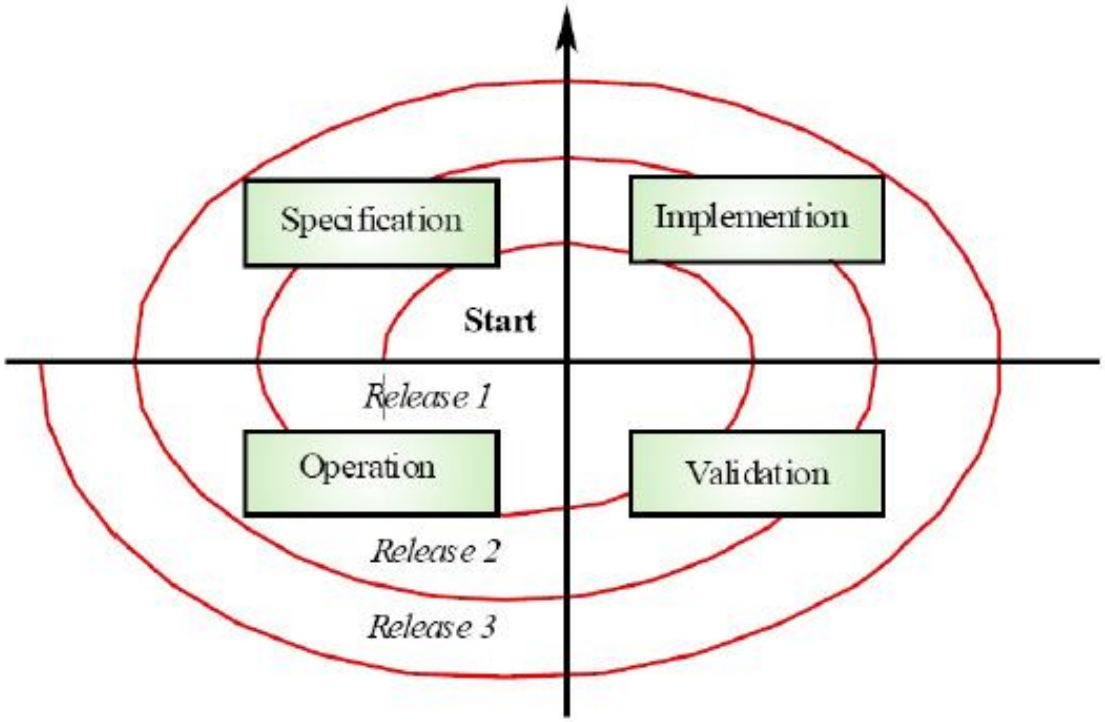
Google docs

TO-DO list

Archivo Editar Ver Insertar Formato Herramientas Formulario Ayuda

	A	B	C	D	E	F
2	Número	Descripción	Prioridad	Asignado a	Estado	Observaciones
3	1	No se muestra la pantalla de opciones del GPS	Baja	Isamu	Arreglado	
4	2	Al pulsar "Exit" en el navegador, la ejecución se interrumpe	Normal	Toni	Arreglado	
5	3	La brújula no se muestra correctamente en determinadas ocasiones	Alta	Doria	En proceso	¡No entiendo porqué sucede!
6	4	Las preferencias de usuario no se cargan al iniciar la aplicación	Alta	Toni	Arreglado	Estaba comentada una línea
7	5	No encuentra los POI en las carpetas del móvil	Normal	Isamu	Pendiente	
8	6	La aplicación no tiene los permisos necesarios para usar BT en S60	Alta	Toni	Arreglado	
9	7	Terminar las traducciones al inglés	Baja	Doria	Arreglado	

En nuestro caso, hemos mantenido una lista colaborativa de casos pendientes de arreglo, de forma que los tres miembros del grupo hemos podido ir asignándonos cada tarea y arreglándolas según surgían. Este patrón se asemeja al denominado **mantenimiento integrado en el proceso de desarrollo**.



A la hora de hacer pruebas se ha puesto de manifiesto la dificultad de hacer debugging en programas para plataformas móviles. Nos hemos encontrado en multitud de ocasiones frente a un contratiempo que se planteaba en el dispositivo móvil que antes no había aparecido durante los test en el emulador.

MANUAL DE USUARIO

1. ANTES DE COMENZAR...

Si usted se está planteando instalar Geomantes en su dispositivo móvil asegúrese de contar con un sistema GPS interno o mediante Bluetooth.

Notas sobre la batería

Le recomendamos que cargue completamente su dispositivo cuando se disponga a usar Geomantes por largos periodos de funcionamiento. Asimismo, le instamos a que si va a hacer uso de la aplicación en el coche, lo haga con un sistema de sujeción adecuado y conectado a la toma de corriente.

★★★★★★★★★★

***ATENCIÓN**: No interactúe con su teléfono móvil mientras se encuentra conduciendo.*

★★★★★★★★★★

Recepción de señal

Si Geomantes tarda más de cinco minutos en encontrar la posición en la que se encuentra, por favor asegúrese de que el dispositivo se encuentra en un espacio abierto y con visibilidad directa hacia el cielo.

2. EL SISTEMA DE MENÚS

Debido al escaso y elemental apoyo gráfico del que dispone J2ME, Geomantes consta de un interfaz gráfico desarrollado " desde cero " para dotar al programa de un aliciente visual superior al de

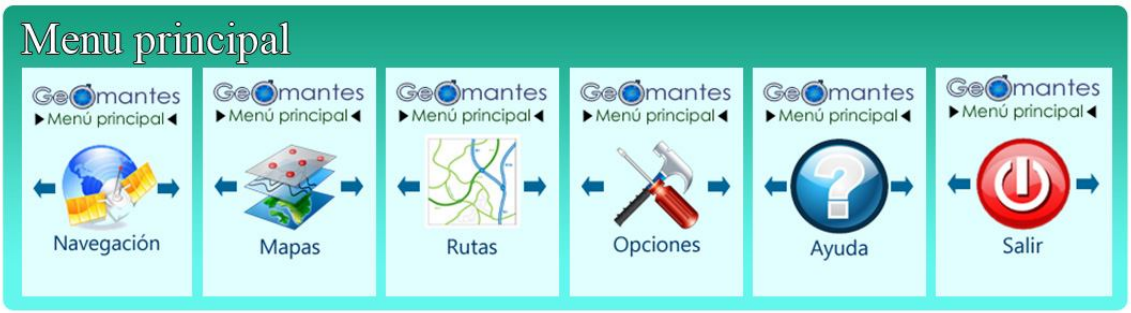
sus competidores. Se ha optado por un sistema de menús dispuesto en forma de árbol de decisión, donde cada opción escogida por el usuario, abre otras nuevas de forma clara e intuitiva.

2.1 La pantalla de bienvenida

Su función principal es la de permitir la carga transparente de información en *background* sin que el usuario lo perciba. En inglés se denomina *Splash Screen*.

2.2 El menú principal

A continuación, se detalla la explicación de la labor de cada pantalla del sistema de menús.



Navegación: Principal funcionalidad del programa. Permite al usuario navegar por distintos tipos de mapa, tanto conectado a un dispositivo GPS con autónomamente, a través de mapas cargados en el móvil o mediante internet y Google Maps.

Mapas: Permite cargar un mapa en la aplicación, o conectarse a través de Google Maps.

Rutas: Muestra una serie de opciones relacionadas con las rutas, como iniciar una grabación, la carga de una previamente almacenada o su reproducción.

Ayuda: Se muestra información para que el usuario pueda navegar con conocimiento a través del programa y no encuentre dificultades.

Salir: Finaliza la ejecución. Antes de devolver la ejecución al sistema del móvil, se ocupa de almacenar toda la información de la aplicación en el sistema RMS para que la próxima vez que se abra Geomantes, la información quede actualizada.

2.2.1. Mapas

Cargar mapa: Abre un navegador de archivos que permite seleccionar un fichero *.map* para que el programa se encarga de gestionar los mapas asociados.

Descargar mapa: Permite almacenar en el móvil mapas generados a través de Google Maps para su futura utilización sin requerir una conexión a internet.

Borrar mapa: Permite eliminar un mapa almacenado en la memoria de nuestro dispositivo.

Modo Google Maps: Al seleccionar esta opción, hacemos que nuestra posición se muestre sobre un mapa cargado desde Google Maps. Es necesario disponer de conexión a internet.

Previsualizar mapa: Permite ver en pantalla de forma gráfica que mapa hemos cargado en memoria.

2.2.2. Rutas

Iniciar grabación: Graba en tiempo real el recorrido que se está realizando para su posible posterior utilización.

Finalizar grabación: Termina la grabación en curso.

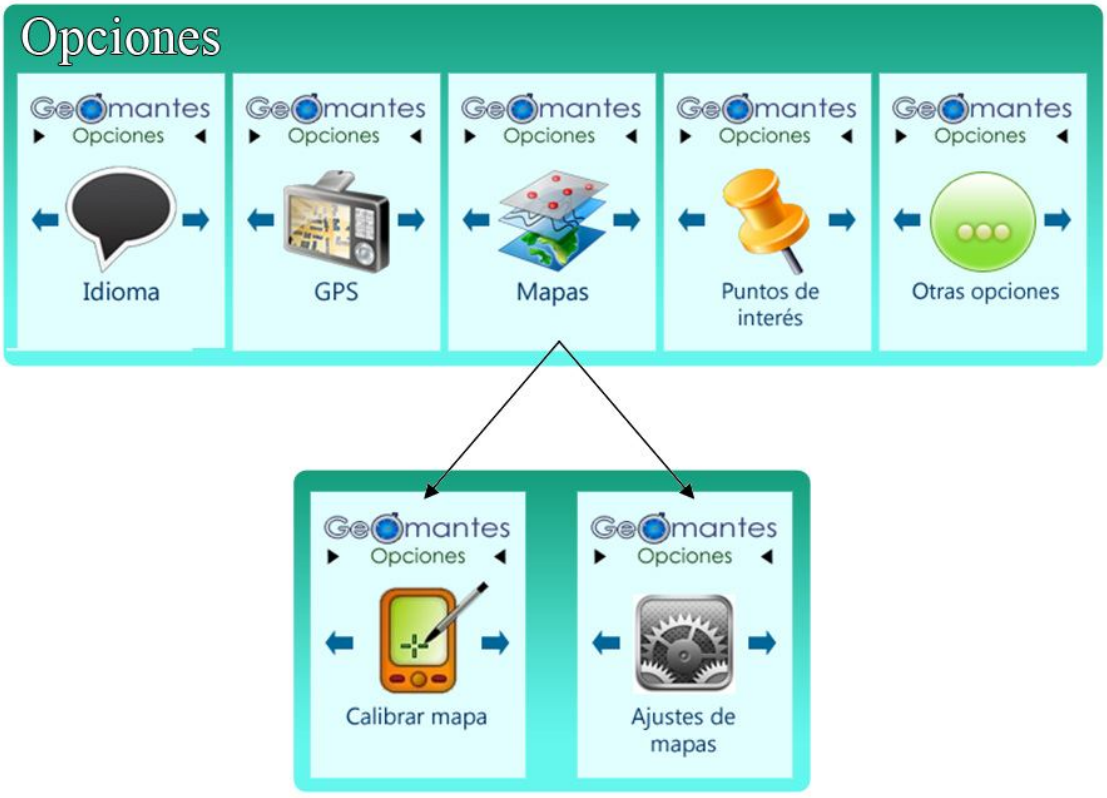
Cargar ruta: Abre un navegador de archivos y permite seleccionar una ruta ya generada. Esto nos servirá para reproducirla o para hacer una navegación guiada.

Descargar ruta: Libera de la memoria la ruta cargada en ese instante.

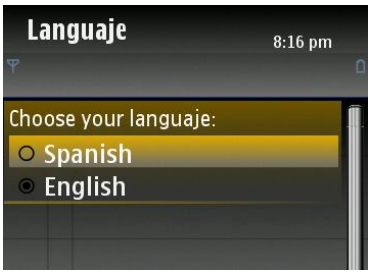
Borrar ruta: Elimina de nuestro móvil la ruta cargada.

Reproducir ruta: Nos permite recrear la ruta cargada mediante un seguimiento de la misma.

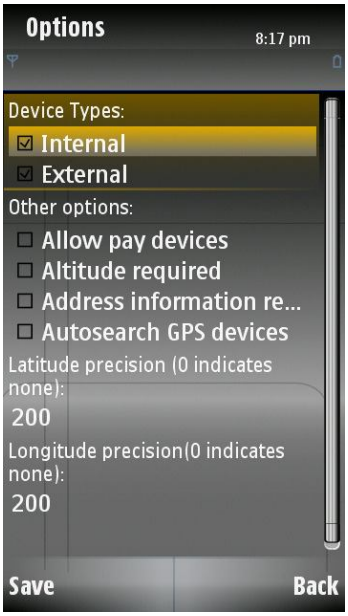
2.2.3. Opciones



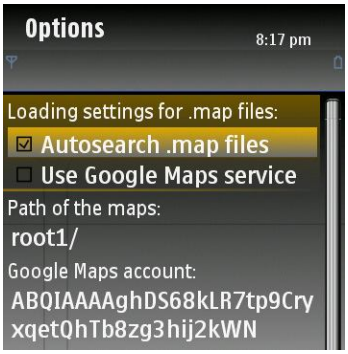
Idioma: Nos permite escoger entre los distintos idiomas soportados por Geomantes.



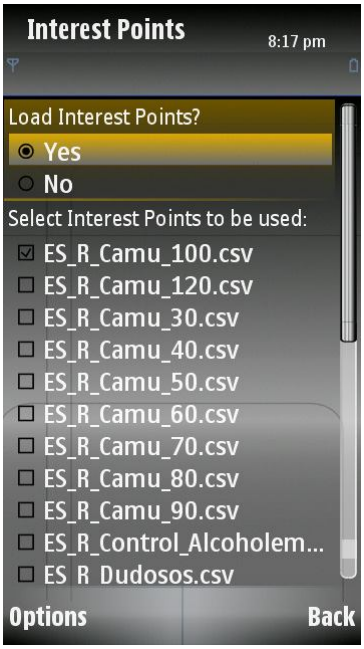
GPS: Nos permite seleccionar entre una serie de opciones asociadas al uso del GPS. Podremos filtrar con qué tipo de dispositivo nos conectamos (externo o interno), y otras diversas opciones como permitir dispositivos de pago, exigir que devuelva la altura o que busque automáticamente un GPS al iniciar la ejecución.



Mapas: Permite gestionar la localización de los mapas en nuestro dispositivo móvil, nuestra cuenta en Google Maps y las opciones de carga.

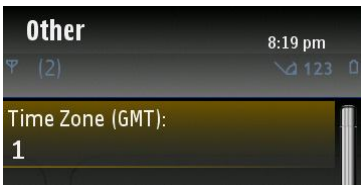


Puntos de Interés: Menú de opciones donde manejamos si mostrar los puntos de interés asociados al programa. Para ello tendremos una carpeta habilitada para dicho propósito. Podremos seleccionar que POI son cargados y cuáles no.



Calibrar Mapa: A través de dos puntos de una imagen, esta funcionalidad nos permite geolocalizar perfectamente un mapa. Para ello necesitaremos o bien un dispositivo GPS o conocer las coordenadas exactas de los puntos seleccionados.

Otras opciones: Aquí se recogen aquellas opciones que no pueden ser catalogadas en ningún otro lugar como el ajuste de zona horaria.



3. LA PANTALLA DE NAVEGACIÓN

1. Panel de información superior
2. Puntos de interés (POI)
3. Posición actual
4. Zona de mapas
5. Brújula
6. Panel de opciones

La pantalla de navegación es donde se produce la navegación propiamente dicha. En ella se dispone la información que puede serle útil al usuario para hacer de la navegación una experiencia más completa, con la importante misión de no empeorar la experiencia de uso.

1. Panel de información superior: Es la zona de la pantalla donde más información se recoge.

    1.1. Velocidad actual.
    1.2. Posición (latitud y longitud).
    1.3. Distancia recorrida.
    1.4. Calidad de la señal GPS.

2. Puntos de interés (POI): Se sitúan sobre el mapa en la posición y nos indican la presencia de, por ejemplo, un radar, un restaurante, una gasolinera... Normalmente llevan asociada una descripción que es

desplegable y está representada por un punto en la esquina superior derecha del icono.

3. Posición actual: Marcan la situación recibida por el GPS o establecida en la ruta ejecutada. Permanece inalterable en el centro de la pantalla.

4. Zona de mapas: Ocupa la gran parte de la interfaz, y es ocupada por el mapa, ya sea cargado localmente desde el móvil o desde Google Maps.

5. Brújula: Indica la orientación calculada con los dos últimos puntos.

6. Panel de opciones: Nos indica las opciones que podemos desplegar en cada momento.

CONCLUSIONES

1. POSIBLES MEJORAS FUTURAS

Como toda aplicación, a Geomantes de le pueden añadir mejoras, por lo que a continuación pasamos a enumerar las que hemos pensado que quizás podrían ser implementadas en un futuro:

1. Mapas Vectoriales: Desde un inicio se pensó en la posibilidad de introducir mapas vectoriales, sin embargo, durante el desarrollo de la aplicación decidimos restarle prioridad y plantearlo como una futura ampliación, principalmente debido a la falta de tiempo. La posibilidad de usar mapas vectoriales daría funcionalidades adicionales a la aplicación, como pueden ser las rutas guiadas con indicaciones de las direcciones a seguir en cada momento. Es por ello que esta sería una de las principales mejoras y sin duda la que mayor repercusión tendría de cara a la experiencia de uso para el usuario.

2. Aumento de compatibilidad: Un objetivo primordial siempre ha sido el de poder llegar al mayor número de usuarios posibles pero a veces es una tarea compleja y todavía se podría seguir trabajando en este ámbito. La falta de estandarización de tecnologías, funciones, capacidades y potencia entre los distintos dispositivos móviles hacen de esta una tarea ardua pero a la vez importante.

3. Mejoras en el rendimiento: En cualquier aplicación para móviles es muy importante que se haga un uso lo más bajo posible de los recursos del dispositivo dadas las limitaciones de estos, siendo más reseñable sobre aparatos antiguos. A su vez es importante por el ahorro de energía, ya que no hay que olvidar nunca que hablamos

de dispositivos móviles que cuentan con una batería y por lo tanto con una autonomía limitada.

4. Difusión, promoción y servicio técnico: Dado que Geomantes es un producto que responde a una necesidad del mercado, uno de los objetivos básicos es prestar un servicio a la gente. Para ello haría falta un plan de promoción por la red, compañeros y conocidos. Además, se podría plantear la posibilidad de ofrecer un servicio técnico al usuario, eventualmente de pago, con el que sufragar los gastos derivados de la propia promoción y desarrollo de Geomantes.

2. LECCIONES APRENDIDAS

Pese a que la experiencia de desarrollar una aplicación como Geomantes ha sido altamente gratificante y enriquecedora para los tres miembros del equipo, la conclusión más evidente que podemos sacar no es otra cosa que una advertencia:

★★★★★★★★★★

Desarrollar aplicaciones para móviles puede desesperar a más de un valiente aventurero.

★★★★★★★★★★

Intentaremos plantearlo en términos porcentuales: Calculamos que más de la mitad del tiempo invertido en el desarrollo de la aplicación se ha dedicado a resolver problemas de compatibilidad, conflictos de tecnologías e intentar acceder a todas las funcionalidades que brindan los móviles de hoy en día.

Por otra parte, todo el periodo de pruebas se ha visto realmente ralentizado por el hecho de que no es posible realizar debugging en el ordenador pese a contar con emuladores. Además, los errores y excepciones que surgen en

el móvil no pueden ser trazados ni detectadas, por lo que se convierte en una tarea realmente laboriosa y un tanto árida.

Pese a la evidencia matemática de que un punto negativo es el que más peso ha tenido en nuestro proceso de aprendizaje a lo largo del año, no podemos dejar de contentarnos al descubrir que un proyecto que no existía ha salido adelante con nuestro trabajo e imaginación. Geomantes suponía un reto para nosotros ya que abarcaba tecnologías inexploradas para nosotros que pondrían a prueba nuestra capacidad de aplicar muchos mecanismos asimilados a lo largo de toda la carrera.

Una parte importante de la carga de trabajo asociada al proyecto ha estado muy condicionada por nuestra capacidad de organización, trabajo en equipo y rapidez de toma de decisiones. Lo que se deduce de todo lo escrito es que al final, y pese a lo que puede parecer inicialmente, el tiempo que se pasa redactando código frente al ordenador no supera un 30% del tiempo total invertido.

Por último, es importante notar las formas más básicas de aprendizaje que hemos adquirido: conceptos como la geolocalización (y todo lo que lleva asociado tras de sí) o la firma de aplicaciones móviles nos eran totalmente ajenos antes de emprender el proyecto. La conclusión más inmediata es que Geomantes nos ha enriquecido en aspectos tanto académicos como personales.

BIBLIOGRAFÍA

[1] http://www.elgps.com/

[2] www.j4me.org/

[3] http://www-users.cs.umn.edu/~czhou/docs/jsr179/lapi/javax/microedition/location/

[4] http://www.mailxmail.com/curso-programacion-juegos-moviles-j2me/almacenamiento-rms

[5] http://es.wikipedia.org/wiki/GPS

[6] http://www.elgps.com/documentos/datum.html

[7] http://synergix.wordpress.com/

[8] http://es.wikipedia.org/wiki/Diagrama_de_casos_de_uso

[9] http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

[10] http://www.chuidiang.com/ood/patrones/modelo_vista_controlador.php

[11] http://www.proactiva-calidad.com/java/patrones/mvc.html

[12] http://es.wikipedia.org/wiki/Singleton

[13] http://www.elrincondelprogramador.com/default.asp?id=45&pag=articulos%2Fleer.asp

[14] http://chuidiang.blogspot.com/2005/12/patrn-singleton.html

[15] http://www.rus-roads.ru/gps/help_ozi/map_file_format.html

[16] http://www.cartesia.org/data/apuntes/cartografia/cartografia-geograficas-utm-datum.pdf

[17] http://www.mapealo.com/Costaricageodigital/Documentos/alfabetizacion/proyeccion_datum.pdf

[18] http://es.wikipedia.org/wiki/WGS84

PALABRAS CLAVE

- GPS
- Geolocalización
- J2ME
- JSR 179
- Bluetooth

Los alumnos que hemos desarrollado este proyecto autorizamos a la Universidad Complutense de Madrid a difundir y a utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria como el código, la documentación y/o el prototipo desarrollado.

Antonio Ariza Guerrero Javier Doria Dulanto Isamu Takebe Heras